REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE -	3. REPORT TYPE AND DATES COVERED FINAL 02 Mar 98 - 01 Mar 00

4. TITLE AND SUBTITLE
Unmanned Air Vehicles

5. FUNDING NUMBERS

DAAG55-98-1-0094

6. AUTHOR(S)
S. Shankar Sastry

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of California - Berkeley

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

ARO 38130.1-CI-RIP

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A major part of our research effort on unmanned autonomous vehicles is the development and fabrication of an aerial platform capable of supporting research on a number of topics, including multi-agent hybrid systems involving sensor fusion, discrete decision making under uncertainty, coordinated mission planning, and distributed control. The foundation of an experimental system on these topics is a dependable autonomous aerial platform that is responsive to requests for basic flight maneuvers such as takeoff, landing, hover, and waypoint navigation. The autonomous aerial platform developed on ARO Grant DAAG55-98-1-0094 at the University of California, Berkeley, consists of reliable aerial vehicles, integrated position and attitude sensors, embedded real-time flight controller and auxiliary computing systems, communication packages, and vision capabilities.

RECEIVED
MAY 3 1 2001
By

14. SUBJECT TERMS

15. NUMBER OF PAGES
9

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

20010619 084

UNMANNED AIR VEHICLES
Final Report DAAG55-98-1-0094
S. Shankar Sastry
Electronics Research Laboratory
University of California, Berkeley, CA 94720

A major part of our research effort on unmanned autonomous vehicles is the development and fabrication of an aerial platform capable of supporting research on a number of topics, including multi-agent hybrid systems involving sensor fusion, discrete decision making under uncertainty, coordinated mission planning, and distributed control. The foundation of an experimental system on these topics is a dependable autonomous aerial platform that is responsive to requests for basic flight maneuvers such as takeoff, landing, hover, and waypoint navigation. The autonomous aerial platform developed on ARO Grant DAAG55-98-1-0094 at the University of California, Berkeley, consists of reliable aerial vehicles, integrated position and attitude sensors, embedded real-time flight controller and auxiliary computing systems, communication packages, and vision capabilities.

SYSTEM ARCHITECTURE

We base our unmanned air vehicle (UAV) system on radio-controlled helicopters. Each helicopter is modeled as a hierarchical hybrid system. The system is inherently hybrid, having to combine continuous control with discrete logic. Hierarchy allows to separate complex global task in a series of simpler, local ones. Each helicopter model consists of three components: (1) a flight management system (FMS) which is responsible for planning and controlling the operation of the UAV. (2) a vision system for the detection and investigation of objects in the environment and (3) the helicopter itself, i.e., the plant to be controlled. As described in Figure 1, the FMS consists of four layers, strategic, tactical, and trajectory planners, and a regulation layer. (1) A Strategic Planner is concerned with the planning and execution of the central UAV mission. It designs a coarse, self-optimal trajectory, which is stored in form of a sequence of waypoints. This layer also takes care of the transition between the points, by acknowledging the completion of a subtask and scheduling the next one. (2) A Tactical Planner is responsible for local obstacle avoidance: it plans a discrete trajectory between the waypoints provided by the Strategic Planner and must modify it online in real time in case of appearance of new obstacles along the previously planned path. To accomplish this, it makes use of data provided by cameras, GPS and internal sensors on position, orientation, linear and angular velocities. (3) A Trajectory Planner interpolates the set of waypoints into a continuous trajectory which the lower layers of the system will have to follow. Such a trajectory will have to be trackable, i.e., compatible with the UAV dynamics. In safety critical situations, the Trajectory Planner might overrule the behavior proposed by the Tactical Planner, and send to the system's lower layers continuous trajectories that correspond to safety maneuvers. The (4) Regulation Layer and Dynamic Layer represent the continuous control part of the system.
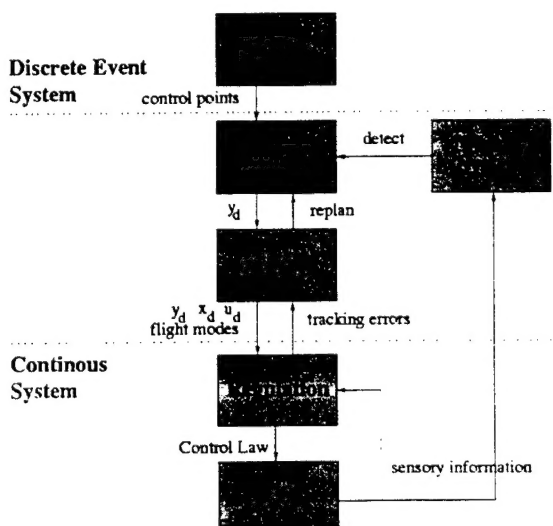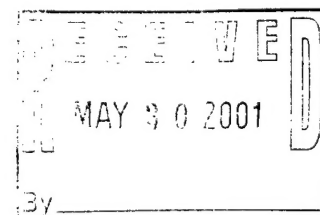

Figure 1: System Architecture

1

ROTORCRAFT UNMANNED AERIAL VEHICLE (RUAV) TESTBED

During the past two years, Berkeley UAV team has constructed and successfully flight tested two rotorcraft-based UAVs (RUAVs). A small UAV capable of hovering, based on a Kyosho Concept 60 hobby-purpose radio controlled helicopter. and a much larger UAV. based on a Yamaha R-50 radio-controlled helicopter, able to perform a wide range of flight maneuvers and waypoint navigation. Two Yamaha R-50 helicopters (10-ft main rotor and 44-lb payload capacity) were acquired on a different grant.

On both the Kyosho Concept 60 and the Yamaha R-50. once hardware and software was integrated with the helicopter. the stabilizing control law for the helicopter airframe was sought. For controller design, the system model valid for hovering flight was identified. Using this model, state-space based linear robust control theory as well as classical control theory was applied for hovering controller design. The designed controllers were validated in simulation and then tested in test flights at the Richmond Field Station. Both controllers showed satisfactory performance in actual test flights and the reliability of the acquired vehicle model in hover was also validated. Once the controller demonstrated satisfactory response, it was integrated into a hierarchical waypoint navigation system along with the introduction of our Vehicle Control Language (VCL). The UAV performed waypoint navigation with great flexibility in a series of real flight tests.

EQUIPMENT PURCHASED ON THE DURIP

We followed the request on the DURIP proposal very closely.The following equipment was purchased on the DURIP:

1. Novatel GPS for the rotorcraft UAV
2. 2 Boeing DQI INS for the rotorcraft UAV INS
3. Yamaha helicopter parts for the rotorcraft UAVs
4. SICK laser range finders for the helicopters and UGVs
5. Computers for modeling, simulation and the ground station and mobile command stations:
   - 2 SGIs for simulation
   - 2 Sun Ultra SPARC stations for smulation
   - 3 IBM laptops for mobile UAV/UGV command stations
   - 3 Dell Workstations for the ground controller

In addition, we paid for materials for fabrication, electronics, communications, wireless LAN and parts for the rotorcraft UAV testbed

UAV Testbed. A UAV is a vehicle integrated with mechanical and electronic components such as airframe, navigation sensors, computers, batteries and other sensors, performing autonomous tasks desirably with minimal intervention by remote human operators. The onboard components can be categorized as follows: (1) flight control computer (FCC), (2) navigational sensors, (3) communication module, and (4) onboard power pack.
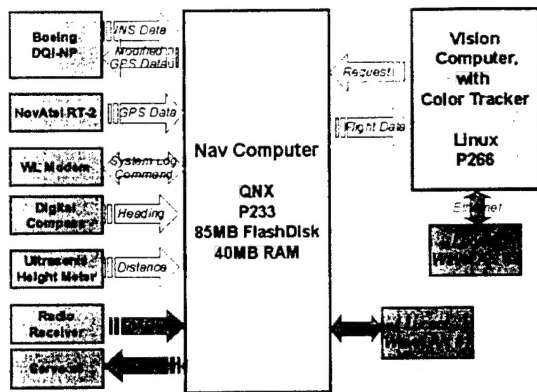
Figure 2. Onboard flight control system structure

Dynamic Model Identification. The acquisition of a high fidelity system model for the vehicle is a crucial step towards the successful design of high-performance flight control system. System identification of an RUAV system provides complexities not encountered in fixed-wing aircraft due to its multi-input multi-output (MIMO), nonlinear characteristics, severe noise and disturbance, and wide flight envelope. Helicopter dynamics can be simplified using a general parametric model consisting of main rotor, tail rotor, fuselage and stabilizer fins. A rigorous approach to obtain full-envelope nonlinear dynamic model, however requires aerodynamic formulae for each aerodynamic component. The resulting dynamic model is unquestionably nonlinear, involving many aerodynamic or mechanical parameters. Among the many unknown parameters, aerodynamic factors such as lift and drag coefficients of blades and fuselage call for experiments using special testing rigs. Other quantities, such as vehicle inertia, while not impossible to measure, also pose significant difficulties.

An alternative approach, to avoid these limitations, is to identify the vehicle model in its entirety using actual flight data. Hence, in our research, LTI model identification was selected and used with the following results. The dynamics of our RUAVs is in general similar to that of the full-size helicopters, while our RUAVs typically show faster responses due to their smaller inertia and faster rotor revolution. Hence their dynamics are artificially retarded by the stabilizer bar mechanism on the main rotor in order for the pilots on ground to maintain remote control. The gyroscopic effects of the stabilizer bar introduce response time delay and damping. The resulting helicopter dynamics shows stable angular rate dynamics in roll and pitch, unlike those without the stabilizer system. This difference requires that the parametric model include stabilizer dynamics. The parametric estimation method requires flight data for each flight mode sampled at a rate sufficiently higher than the targeted response speed. The hover model was chosen as the base model for hover and low-velocity flight up to 5 m/s. The control input and the vehicle output, i.e., position, translational velocity, attitude angles and angular velocity, were measured using PWM signal by circuit and high-accuracy onboard navigation sensors, respectively, and downloaded to the ground computer. After preprocessing, the suitable identification algorithm is applied to the flight data. The resulting system model is a six degree-of-freedom linear rigid body helicopter model with first-order servorotor dynamics.

Control Law Design. The RUAV platform needs to be stabilized and controlled in order to maneuver through a given series of waypoints. For this purpose, an onboard real-time controller needs to be designed into the feedback loop. This can be done either by using classical control theory or modern state-space control theories. The method widely used industry or military applications is a classical SISO approach, due to its simple and intuitive nature, and more importantly, its satisfactory performance proven in actual flight tests.
In Figure 3, the structure of multi-loop SISO classical compensator is shown. This simple structure of classical approach allows simple, but very effective control algorithms. In cruise mode, the velocity and attitude loops are closed for stabilization and tracking. When hovering over a certain spot is required, the outmost loop for position feedback is closed along with the inner loops. A series of experiments has been performed using the proposed controller on theYamaha R-50 based UAV. During repeated experiments, the attitude/velocity controller has shown stable operation even when the helicopter stays on the ground. Therefore, more accurate take-off and landing can be achieved by activating the attitude/velocity controller even before the helicopter takes off from the ground. When operated manually, the pilot engages the attitude/velocity controller using a switch on the transmitter and then lifts the helicopter off from the ground. At this time, only steady heave reference command is

3

given. Once the helicopter reaches the desired altitude. the hovering controller (the position/velocity/attitude loop controller) is activated.

Figure 4 shows our R-50 UAV in automatic hover. The RUAV showed a stable response over two minutes with ±0.5 m accuracy in x and y direction. Roll. pitch and translational velocities in x and y directions were well integrated together. The altitude regulation shows an outstanding performance with only ±0.1 m error and the heading regulation with only ±3 degrees error.
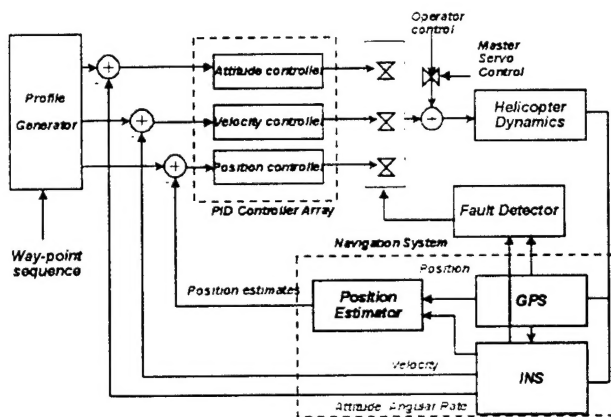

Figure 3. The SISO multi-loop controllers

Figure 4. Yamaha R-50 in flight experiment

μ- Synthesis Controller Design. As an alternative to the classical approach. we applied the modern MIMO linear control theory to the helicopter control problem. Due to the inherent cross-coupling of the rotor dynamics, MIMO control algorithms are considered more desirable than SISO controllers. With the emphasis on robustness, among many MIMO control theories. μ-synthesis control theory explicitly accounts for structured uncertainty of the system. μ-synthesis approach also allows to describe the sensor noise model and to deign a controller satisfying the performance criterion in the presence of the uncertainty and sensor noise. These features are particularly attractive because the controller must perform stabilization of the nonlinear unstable helicopter system in the presence of uncertain and/or poorly known system dynamics along with severe disturbance and sensor noise. Preliminary testing of μ-synthesis approach on the roll and pitch regulation problem showed good results.

VEHICLE GUIDANCE USING VEHICLE CONTROL LANGUAGE

Vehicle Control Language (VCL) is a framework for RUAV guidance, which has the hierarchical structure shown in Figure 5. This approach allows the abstraction of a case-by-case guidance sequence as well as the integration of the low-level vehicle control and the abstract-level mission planning. Using this framework, the onboard autopilot system performs any given feasible mission without any change of the program of onboard software. VCL has well-defined semantics based on finite state automata. which makes it easy to see how the system will react. The sequence of motion commands is described in a form of a script language understandable both to humans and to the decoder software. VCL module consists of user interface part on ground station, language interpreter and sequencer on the UAV side.

When a mission is determined. the ground or onboard operator specifies a sequence of waypoints with their attributes such as the type of waypoint. heading, velocity, etc. When completed. the VCL is uploaded to the RUAV control system and then executed sequentially. The VCL execution module (VCLEM) selects the proper controller, depending on the flight mode. and generates the reference command. VCLEM monitors the vehicle trajectory and determines if one sequence is finished or not. It also monitors the vehicle status for possible troubles in sensors or the vehicle itself. If an error is detected, a fault detection algorithm is activated and the proper error handling measure will be exercised. In the worst case, the VCL releases the vehicle control to the ground-based test pilot. Example keywords and syntax of VCL are shown in Figure 6. Currently the VCL vocabulary covers all the basic maneuvers. and it will be expanded as more flight modes are realized.
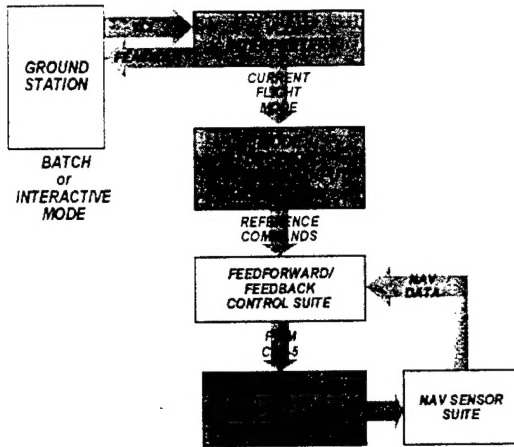
Figure 5. Hierarchical architecture of VCL processing

```
TakeoffTo <coord>{abs,rel} : perform autonomous take-off to certain target point

Hover <coord>{abs,rel} {heading=<heading>{deg,rad}} <duration> {sec,min}

 : hover with given heading angle for given tim



FlyTo <coord>{abs,rel} {vel=<velocity>{mps,kmps,fps,knots,mph}}{passby,stopover} {autoheading,
heading=<heading>{deg,rad}}

: cruise to certain waypoint stopping over or passing by
```

Figure 6. Syntax of Vehicle Control Language

The proposed VCL processor is implemented in onboard flight software. The software is first validated in MATLAB/Simulink and then tested in real flight condition. Fig 6 shows the sample VCL code describing a sweeping path of a certain area. This code is tested in real flight and the result is shown in Fig 7. As this graph suggests, the VCL processor could execute the requested maneuver with acceptable accuracy.


```
Hover  (0,0,0)rel heading=270deg duration=10sec;
FlyTo  (0,-5,0)rel vel=0.5m/s stopover autoheading;
Hover  (0,0,0)rel heading=0deg duration=10sec;
FlyTo  (5,0,0)rel vel=0.5mps stopover autoheading;
Hover  (0,0,0)rel heading=270deg duration=10sec;
FlyTo  (0,-5,0)rel vel=0.5m/s stopover autoheading;
Hover  (0,0,0)rel heading=180deg duration=10sec;
FlyTo  (-5,0,0)rel vel=0.5mps stopover autoheading;
Hover  (0,0,0)rel heading=270deg duration=10sec;
FlyTo  (0,-5,0)rel vel=0.5m/s stopover autoheading;
Hover  (0,0,0)rel heading=0deg duration=10sec;
FlyTo  (5,0,0)rel vel=0.5mps stopover autoheading;
Hover  (0,0,0)rel heading=270deg duration=10sec;
```
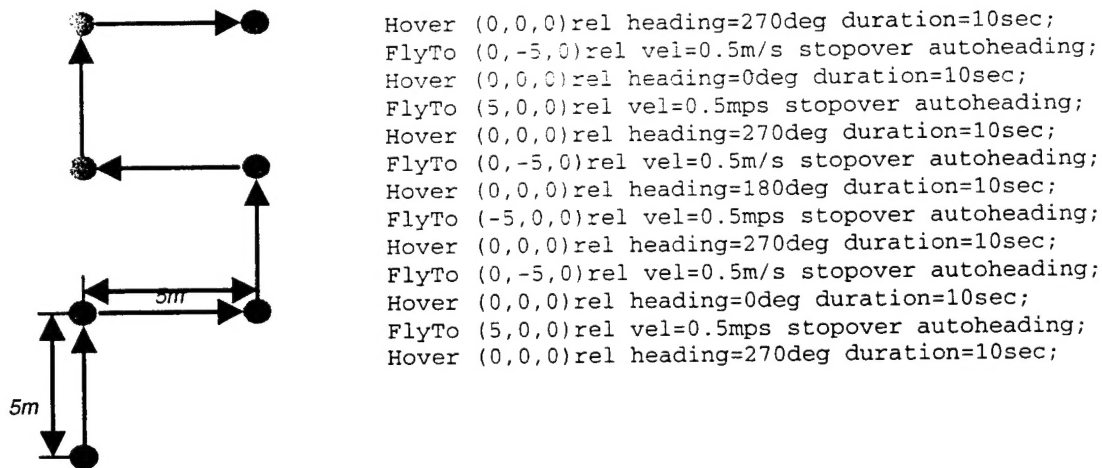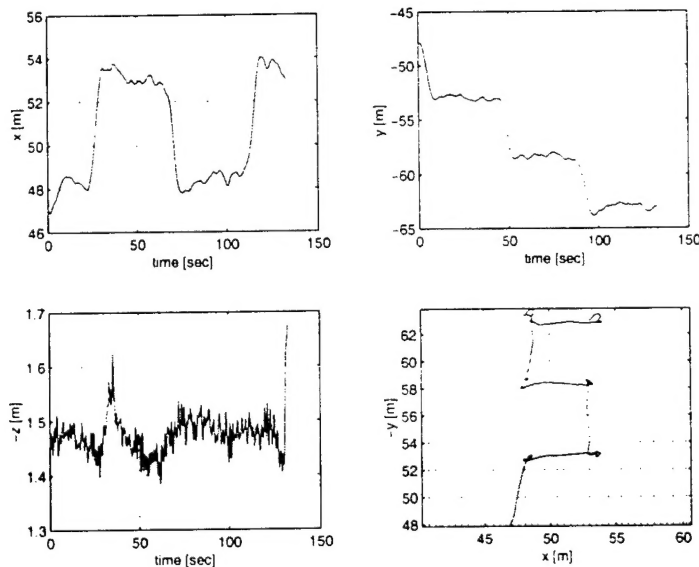
Figure 7. Sample VCL code for flight experiment

Figure 8. Experimental result of sample VCL code

VISION BASED NAVIGATION FOR AN UNMANNED AERIAL VEHICLE

We are developing a system for autonomous navigation of UAVs based on computer vision. A UAV is equipped with on-board cameras and each UAV is provided with noisy estimates of its own state data, coming from GPS/INS. The mission of the UAV is low altitude navigation from an initial position to a final position in a partially known 3-D environment while avoiding obstacles and minimizing path length. We use a hierarchical approach to path planning. We distinguish between a global offline computation, based on a coarse known model of the environment and a local online computation, based on the information coming from the vision system. A UAV builds and updates a virtual 3-D model of the surrounding environment by processing image sequences and fusing them with sensor data. Based on such a model the UAV will plan a path from its current position to the terminal point. It will then follow that path, getting more data from the onboard cameras, and refining the map in real time.

6

Without total knowledge of the environment an agent can only plan a path which is optimal with respect to its knowledge at the time of planning. We use a hierarchical approach to path planning, with different paths designed at different time and space scales, and based on multi-resolution environmental models. We distinguish between a global offline computation, based on a coarse model of the environment and a local online computation, based both on a detailed model and on the information coming from the vision system. The UAV makes use of an a priori, inaccurate, knowledge of the terrain to plan an initial, coarse path. A few waypoints are selected based on the desired objective. We use wavelets to filter the map to the desired level of detail. At this stage, the planning is performed deterministically. We use standard optimization algorithms for shortest path computation, such as Djikstra or A*.

On the other hand, in-flight navigation mainly depends on the information gathered by the vision system. We propose a probabilistic approach to online path planning, for a number of reasons: first of all, because of the inevitable uncertainty of measurements from the sensors; secondly, for the intrinsic uncertainty of an unknown surrounding environment; and finally, the structure of reasoning of any (biological or artificial) intelligent system is naturally probabilistic—whenever a decision has to be taken, the costs or gains that all possible choice simply are "weighed" in probabilistic terms, and the decision that is more "likely" to yield maximum gain is taken. The surrounding three-dimensional environment is divided into cells, and each of the cells is assigned with a probability of occupancy. We will call such probability function a "risk map"—a risk map value close to one indicates high risk (presence of an obstacle), while a value close to zero denotes low risk (no obstacle). The UAV has an initial knowledge of the surrounding environment through an a priori risk map assigned from the mission planner. However, such a risk map will be refined by each UAV independently during navigation exploiting sensor data (i.e., multiple image sequences and state data containing UAV's position, orientation, velocity, etc.). Processing multiple image sequences and integrating such information with other sensor readings allows a UAV to estimate the distance between itself and the obstacles, and provide a measure of the uncertainty of the estimates (in terms of error variances). Using all past information in an "optimal" way, the UAV is able to refine its virtual map of the environment, and thus obtain a model that is more accurate and up-to-date.

Given a probabilistic model of the environment, path planning can be performed using dynamic programming techniques to plan a discrete path, as a sequence of adjacent cells. Each cell corresponds to a state of a stochastic transition system, and a cost is assigned to each state transition. The final objective is to minimize the total expected cost. The next section will describe our approach to path planning. The last section is devoted to conclusion and comments.

Path Planning. In this section we shall describe our approach to navigation. We design both a global offline and a local online navigation scheme.

Global Navigation: the Strategic Planner. In 3D navigation the choice of an appropriate model for the surrounding environment is crucial. In our design we make extensive use of Digital Elevation Models (DEM). Recent advances in laser technology have provided us with an extensive coverage of earth surface with extreme level of accuracy. The basic idea consists in gridding the surface and assigning an altitude to each cell in the grid. The gridding is up to 1m with accuracy in the range of centimeters. These models are widely used in Earth Sciences. In our approach we manipulate these models and use them at different levels of resolution.

Algorithm and Results. We employ Dijkstra's algorithm to find the shortest path on the transformed grid. Our cost functionresults from the sum of three factors, appropriately scaled. Every cell point has a cost associated with it.

$$C(i, j) = \alpha_1 c_d (i, j) + \alpha_2 c_e (i, j) + \alpha_3 c_h (i, j)$$

where C_d, C_e, C_h are costs associated with distance to goal, roughness of terrain, flight altitude respectively. In our scenario costs are associated with distance to goal, roughness of terrain, and flight altitude respectively. The UAV looks for the shortest path on a smooth part of the terrain with low altitude. Figure 9 shows a typical outcome of this algorithm. In the figure the waypoints have been interpolated.

Figure 9: Sample Path on a DEM terrain model.
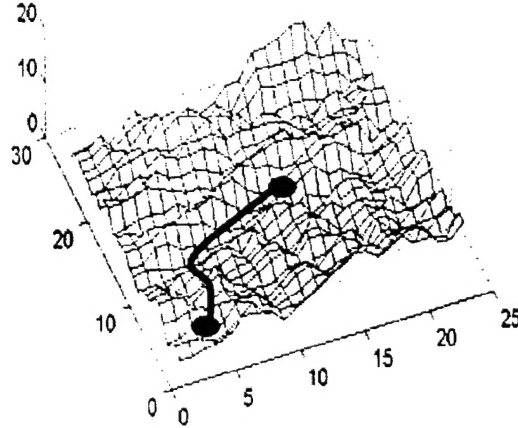
Local Navigation: The Tactical Planner. In this section we shall focus on the problem of vision-based local obstacle avoidance, which is the task of the Tactical Planner. Once a set of waypoints has been provided by the Strategic Planner, what the Tactical Planner roughly has to do is connecting them with a discrete trajectory, i.e. a set of points in three dimensional space which will successively be interpolated into a continuous, trackable trajectory by the Trajectory Planner; the following subsections will clarify this idea. We shall assume that the UAV is provided with an on-board computer, one or more cameras, a GPS system and sensors that give instantaneous, noisy estimates of the agent's orientation in space and (three-dimensional) linear and angular velocities.

Local Path Planning using Dynamic Programming. Using vision information the Tactical Planner is able to build and update the risk map. We will now briefly illustrate a novel algorithm that, given the risk map, provides a sequence of grid points (i.e. a discrete path) connecting the two way-points previously given by the Strategic Planner in a way that detected obstacles are avoided and path length is minimized. Such algorithm is based on Dynamic Programming techniques. We associate a state to each grid point (i.e. state space S coincides with grid G), and we assume that from any state we may move, in one step, to any of its 26 neighbors in the three-dimensional grid; let U be the 26-element control space. We now associate a *cost* function

$c : S^2 \times U \to \Re : (s_i, s_j; u) \mapsto c(s_i, s_j; u)$ to each state pair (s_i, s_j) (i.e. to each state transition) and control u.

We will minimize, with respect to all possible control policies $g : S \to U$, the total expected cost:

$$J^g(s) := \lim_{N \to \infty} E[\sum_{k=0}^{N-1} \alpha^k c(s(k), s(k+1); g(s(k))) \mid s(0) = s].$$

where s is the starting state and $0 < \alpha < 1$; the *optimal policy* is given by the unique solution to Bellman's equation. For our specific path-generation problem we defined cost as a linear combination of three (or more) terms. The first term is proportional to the value the risk map assumes in state s_i (if such value is higher than a certain threshold, say 0.8, then cost is set to a very high value, or infinity); this way trajectories that avoid obstacles will have a lower cost. The second term is proportional to the length of the path connecting states s_i and

s_j, so that globally shorter paths will have a lower cost than others. The third term associates a lower costs to states at a certain altitude from ground, so that the agent is pushed to fly at those altitudes rather than at more costly ones in order to achieve a lower total cost. Finally, we could assign a *gain* (a negative cost) to those areas where the risk map assumes values that are close to 0.5 (maximum entropy), i.e. unknown areas. Thus the agent would be attracted towards unexplored areas—such exploration might yield useful information about obstacle presence (or absence), and allow the Tactical Planner to generate a "better" path. In fact, cost is the translation into mathematical terms of the *task* we want our agent to perform. For example, if we wanted our agent to reach its destination along a known trajectory (and avoid obstacles at the same time), we would just need to add to our cost function a term that is proportional to the distance between each state and the fixed trajectory. This situation is referred to as exploration-exploitation tradeoff, where the first term refers to exploitation of current information while some exploration could increase such knowledge in order to plan a better path. The theory of Dynamic Programming provides fast and efficient techniques for finding approximate solutions to Bellman's equation (which is nonlinear), such as the *value iteration* and the *policy iteration* methods, which we successfully applied to our specific problem. Through computer simulation, we were able to obtain (in real-time) discrete trajectories connecting the first waypoint to the second one, that avoided obstacles and, at the same time, minimized global path length.

Conclusions. Offline computation exploits the a priori knowledge about the environment, providing an initial guess about the optimal route. Online computation exploits the information provided by the vision sensor, capable of sensing the environment. The choice of a probabilistic sensor model and, as a consequence, of a probabilistic online path planning scheme is the most appropriate to capture the natural uncertainty typical of every sensing process. Our approach is hierarchical. Local replanning is performed on fixed horizon, reducing the computation load to guarantee real-time specifications and scalability to maps of any dimension.